Measuring MPI Send and Receive Overhead and Application Availability in High Performance Network Interfaces

Douglas Doerfler and Ron Brightwell

Center for Computation, Computers, Information and Math
Sandia National Laboratories[1]
Albuquerque, NM 87185-0817
{dwdoerf, rbbrigh}@sandia.gov

Abstract. In evaluating new high-speed network interfaces, the usual metrics of latency and bandwidth are commonly measured and reported. There are numerous other message passing characteristics that can have a dramatic effect on application performance that should be analyzed when evaluating a new interconnect. One such metric is overhead, which dictates the networks ability to allow the application to perform non-message passing work while a transfer is taking place. A method for measuring overhead, and hence calculating application availability, is presented. Results for several next-generation network interfaces are also presented.

Keywords: MPI, Overhead, Availability, High Performance Computing, High Speed Networks.

1 Introduction

Scaling efficiency of parallel applications in many instances depends on the ability to overlap communication with computation. If there is sufficient computation to overlap with communication, the application becomes insensitive to the bandwidth provided by the network. Overlap is also beneficial for inherently communication bound codes. In this instance the overhead of preparing the next messages can be overlapped with the transmission of the messages already in the send queue. In MPI application codes, the non-blocking send and receive calls are the primary means of achieving overlap. Unlike other MPI communication metrics, e.g. latency and bandwidth, there is a lack of readily available open-source micro-benchmarks that measure MPI overhead for non-blocking calls. This paper presents a method for measuring overhead and application availability and then applies this method to

[1] Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04- 94AL85000.

several current state-of-the-art high-performance network interfaces. It is not within the scope of this paper to explain why some interconnects and protocols provide low overhead and high availability.

2 Method

There are multiple methods an application can use to overlap computation and communication using MPI. The method assumed by this paper is the post-work-wait loop using the MPI non-blocking send and receive calls, MPI_Isend() and MPI_Irecv(), to initiate the respective transfer, perform some work, and then wait for the transfer to complete using MPI_Wait(). This method is typical of most applications, and hence makes for the most realistic measure of a microbenchmark. Periodic polling methods have also been analyzed [1], but that particular method only makes sense if the application knows that progress will not be made without periodic MPI calls during the transfer. Overhead is defined to be [2]:

> *… the overhead, defined as the length of time that a processor is engaged in the transmission or reception of each message; during this time, the processor cannot perform other operations.*

Application availability is defined to be the fraction of total transfer time[2] that the application is free to perform non-MPI related work.

$$\text{Application Availability} = 1 - (\text{overhead} / \text{transfer time}) \qquad \textbf{(1)}$$

Figure 1 illustrates the method used for determining the overhead time and the message transfer time. For each iteration of the post-work-wait loop the amount of work performed (work_t), which is overlapped in time with the message transfer, increases and the total amount of time for the loop to complete (iter_t) is measured. If the work interval is small, it completes before the message transfer is complete. At some point the work interval is greater than the message transfer time and the message transfer completes first. At this point, the loop time becomes the amount of time required to perform the work plus the overhead time required by the host processor to complete the transfer. The overhead can then be calculated by measuring the amount of time used to perform the same amount of work without overlapping a message transfer and subtracting this value from the loop time.

The message transfer time is equal to the loop time before the work interval becomes the dominant factor. In order to get an accurate estimate of the transfer time, the loop time values are accumulated and averaged, but only those values measured before the work interval starts to contribute to the loop time. These values used in the average calculation are determined by comparing the iteration time to a given threshold (base_t). This threshold must be set sufficiently high to avoid a pre-mature stop in the

[2] Per the MPI non-blocking call definitions, the MPI_Wait() call only signifies that for a send the buffer can be reused and for a receive the data can be accessed in the receive buffer [3].

accumulation of the values used for the average calculation, but not so high as to use values measured after the work becomes a factor. The method does not automatically determine the threshold value. It is best to determine it empirically for a given system by trying different values and observing the results in verbose mode. A typical value is 1.02 to 1.05 times the message transfer time.

Figure 1 also shows an iteration loop stop threshold (iter_t). This threshold is not critical and can be of any value as long as it is ensured that the total loop time is significantly larger than the transfer time. A typical value is 1.5 to 2 times the transfer time. In theory, the method could stop when the base_t threshold is exceeded, but in practice it has been found that this point can be too close to the knee of the curve to provide a reliable measurement. In addition, it is not necessary to calculate the work interval without messaging until the final sample has been taken.

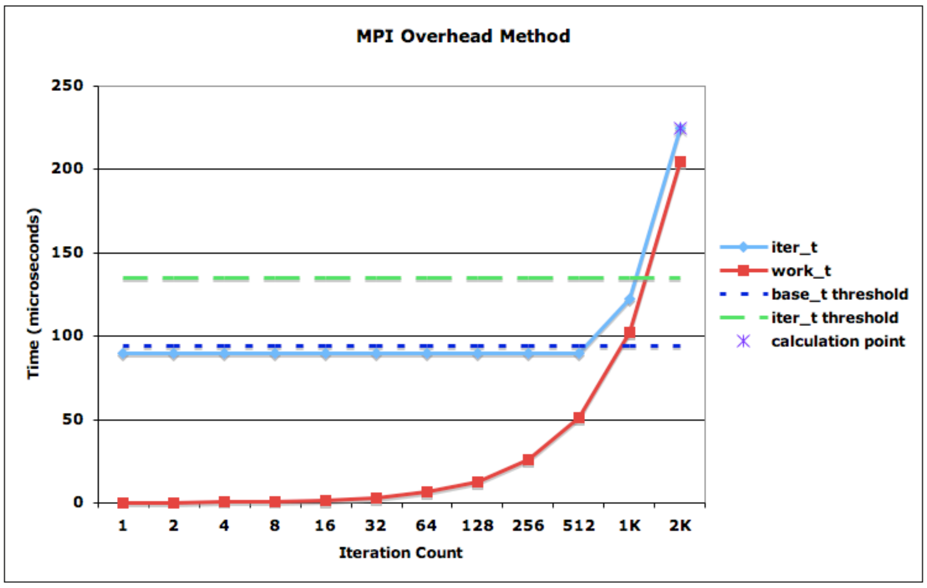

Fig. 1. A conceptual illustration of the post-work-wait loop time (iter_t) of a given message size for each iteration of the algorithm, with the work performed (work_t) increasing for each iteration. The message transfer time calculation threshold (base_t) and the iteration stop threshold (iter_t) are also shown along with the point at which the overhead calculation is taken.

3 Platforms

Overhead and availability was measured on a variety of platforms, summarized in Table 1. All of the platforms except Red Storm are Linux clusters using the respective vendor's commercial software stacks. The Thunderbird cluster's MPI software stack has been modified and parameters have been set to reduce the memory required by the MPI stack at a scale of several hundred to a thousand processes. These modifications do affect the real-world application performance, but it is unknown how those modifications affect the MPI overhead microbenchmark used in this analysis. The Red Storm platform uses the Catamount lightweight kernel [4], with low-level communications implemented using the Portals API [5]. All of the platforms use MPICH 1.x for their implementation of MPI, although several of these implementations have been optimized for their respective network interface. In particular, many vendors have optimized the collective communication routines. The Quadrics software stack uses a patched kernel, which allows optimizations benefiting overhead and host availability performance.

Table 1. Overview of Test Platforms

	Red Storm	***Thunderbird***	***CBC-B***	***Odin***	***Red Squall***
Interconnect	Seastar 1.2	InfiniBand	InfiniBand	Myrinet 10G	QsNetII
Manufacturer	Cray	Cisco/Topspin	PathScale	Myricom	Quadrics
Adaptor	Custom	PCI-Express HCA	InfiniPath	Myri-10G	Elan4
Host Interface	HT 1.0	PCI-Express	HT 1.0	PCI-Express	PCI-X
Programmable coprocessor	Yes	No	No	Yes	Yes
MPI	MPICH-1	MVAPICH	InfiniPath	MPICH-MX	MPICH QsNet

4 Results

From a practical perspective, application availability is usually not a concern for small message sizes, as there is little to be gained trying to overlap computation with communication when transfer times are relatively small. Most applications will only try to overlap computation when they know the message size is sufficiently large. However, as an academic exercise, it still may be interesting to view availability for a small message as it provides information on how an interface's characteristics change at a protocol boundary, such as the switch from a short message protocol to a large message protocol. If an application writer is trying to optimize to a given platform, he/she may want to know where the protocol boundaries are and modify the code to better suit the platform. Overlap may also be beneficial to codes that need to send multiple small messages at a time. In this case, overlap allows preparation of the next message to be put in the queue while the messages already in the queue are being transmitted. However, this is the message throughput metric and is not within the scope of this study.

Figure 2 illustrates the MPI_Isend() overhead as a function of message size for the platforms tested[3]. Figure 3 shows application availability. The overhead for the Red

[3] Note that this figure uses a logarithmic axis for overhead.

Storm, Odin (Myri-10G) and Red Squall (Elan4) interconnects is relatively constant for all message sizes. As such, application availability increases with message size until it is nearly 100% for large message transfers. The Thunderbird (InfiniBand) and CBC (InfiniPath) interconnects show a high overhead for large message transfers, with a corresponding drop in application availability. It should be noted that the InfiniPath network has a relatively low overhead for small transfers, which allows for that interconnect to achieve its high, advertised message throughput rate.

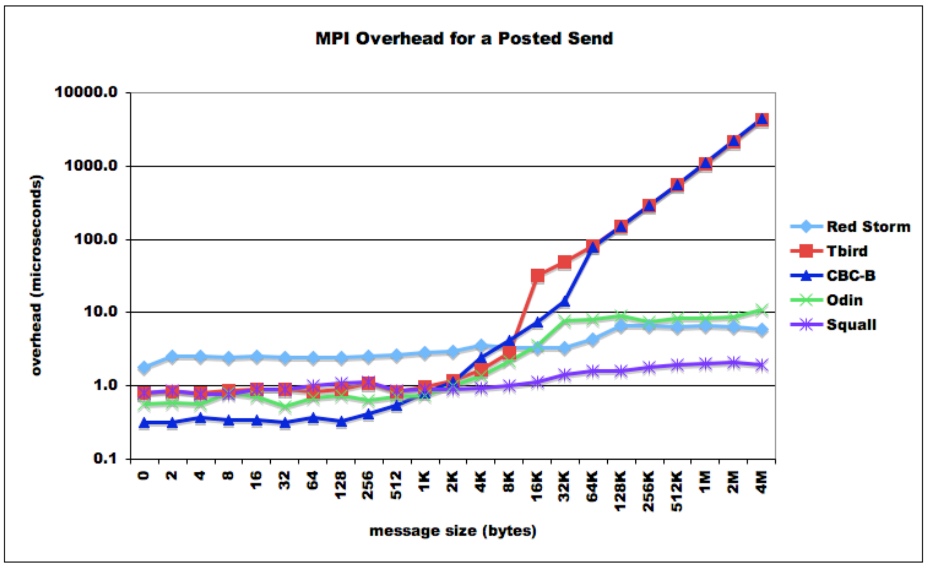

Fig. 2. Overhead as a function of message size for MPI_Isend().

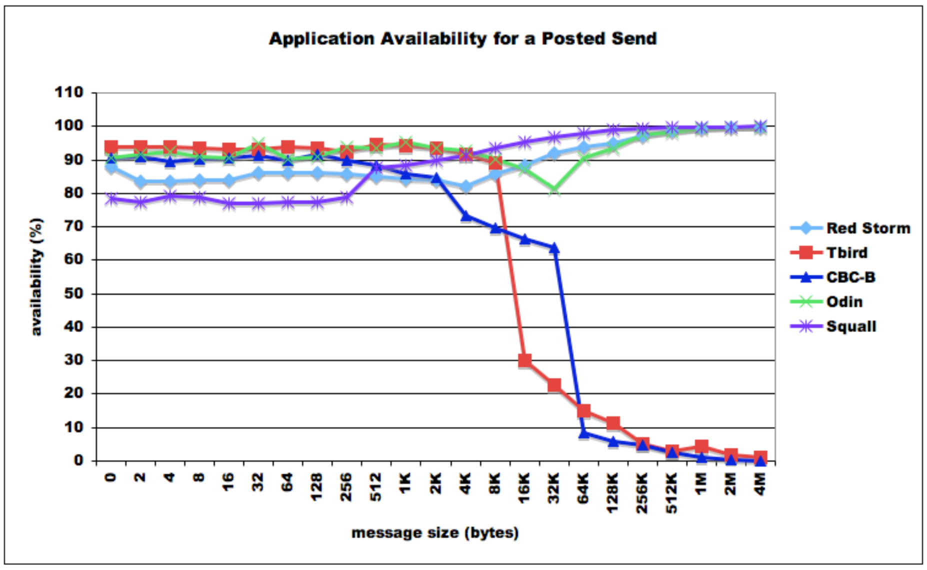

Fig. 3. Application availability as a function of message size for MPI_Isend().

MPI receive performance is charted in Figures 4 and 5. In general, receive performance is similar to the send performance for all of the interconnects tested. The Odin (Myri-10G) cluster does exhibit a more noticeable drop in application availability until the 32K byte message size, which is presumably a protocol boundary. After this point availability increases to an asymptotic value of 100%.

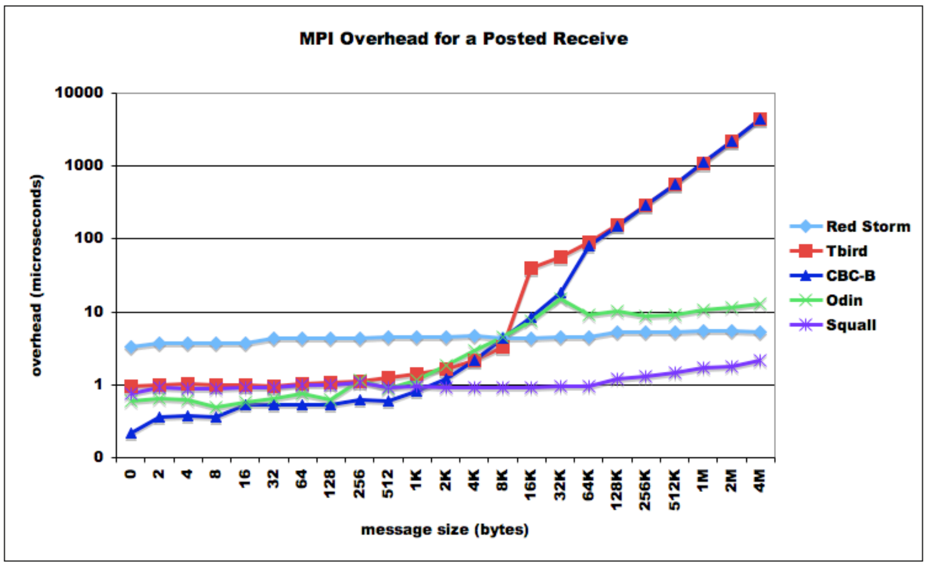

Fig. 4. Overhead as a function of message size for MPI_Irecv().

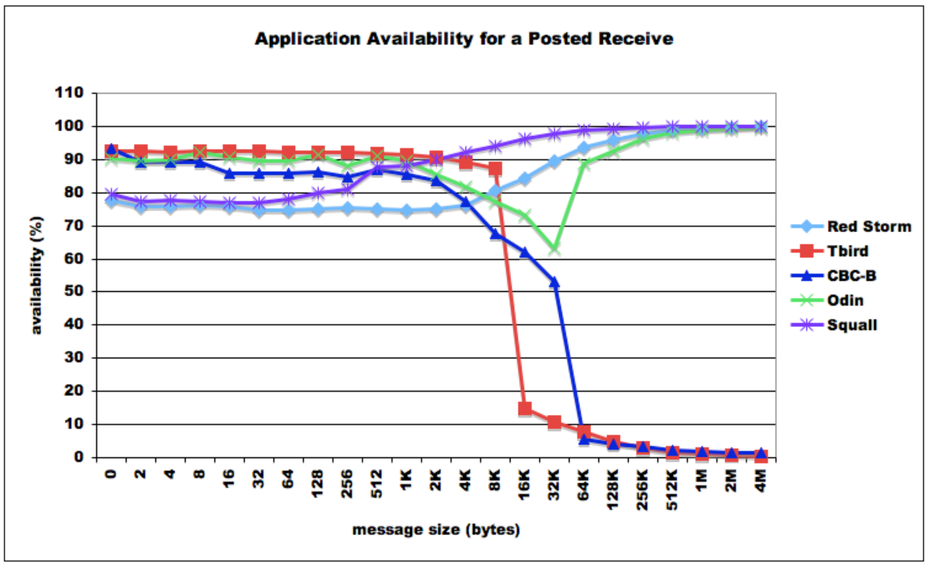

Fig. 5. Application availability as a function of message size for MPI_Irecv().

5 Related Work

A significant amount of prior work has been done to measure and study the effect of overhead on application performance [1], [6], [7], [8] and [9]. Lawry [1] analyzes application availability, but the analysis and results are for a fixed message size and the results are a function of the polling interval. The other previous work does not quantify the overhead as a function of message size, but rather looks at its effect on application performance. An additional contribution of this paper is a comparison of overhead results for relatively new networking technologies, such as Red Storm's SeaStar, Pathscale's InfiniPath, and Myricom's Myri-10G.

6 Conclusion

Simple ping-pong micro-benchmarks do not accurately capture all of the capabilities of a high-performance network. Host overhead and the ability to overlap computation with communication are important performance characteristics that can have a direct impact on an application's scalability. Two networks that have similar latency and bandwidth performance can vary significantly in their ability to provide overlap.

This paper presented a method for measuring overhead and application availability for high-speed networks using MPI and then applied the method to five test platforms, each with a different network interface. Performance for MPI send and MPI receive operations was presented. In general, the send and receive characteristics for a given interconnect were similar. The Red Storm, Odin (Myri-10G) and Red Squall (Elan4) platforms demonstrated a relatively small overhead as a function of message size, and thus showed high application availability for all message sizes. The CBC (InfiniPath) platform demonstrated excellent small message overhead, but for large messages overhead increased linearly with message size and application availability was very low. The Thunderbird (InfiniBand) cluster demonstrated good small message overhead, but like the CBC cluster large message overhead is high and application availability is low.

7 Future Work

It is the intent of the authors to make the source to the code used in this study generally available and downloadable from an open web site, with the hope that this will allow overhead and application availability to become a common micro-benchmark used in the evaluation of interconnects. We also expect that this will encourage contributions from the community to make the code more robust and accurate.

8 References

1. W. Lawry, C. Wilson, A. Maccabe, R. Brightwell. COMB: A Portable Benchmark Suite for Assessing MPI Overlap. In *Proceedings of the IEEE International Conference on Cluster Computing (CLUSTER 2002)*, p. 472, 2002.
2. D. Culler, R. Karp, D. Patterson, A. Sahay, K. E. Schauser, E. Santos, R. Subramonian and T. von Eicken. LogP: Towards a Realistic Model of Parallel Computation. In *Fourth ACM SIGPLAN symposium on Principles and Practice of Parllel Programming,* pp. 262-273, 1993.
3. M. Snir, S. W. Otto, S. Huss-Lederman, D. W. Walker, J. Dongara. MPI: The Complete Reference. p. 52, The MIT Press, Cambridge, Massachusetts, 1996.
4. S. Kelly, R. Brightwell. Software Architecture of the Light Weight Kernel, Catamount. In *Proceeding of the 47th Cray User Group (CUG 2005)*, 2005.
5. *Portals API*, http://www.cs.sandia.gov/Portals.
6. R. Martin, A. M. Vahdat, D. E. Culler, T. E. Anderson. The Effects of Communication Latency, Overhead, and Bandwidth in a Cluster Architecture. In *Proceedings of the International Symposium on Computer Architecture,* 1997.
7. D. Culler, L. T. Liu, R. P. Martin, C. O. Yoshikawa. Assessing Fast Network Interfaces. *IEEE Micro*, pp. 35-43, Feb., 1996.
8. C. Bell, D. Bonachea, Y. Cote, J. Duell, P. Hargrove, P. Husbands, C. Iancu, M. Welcome, K. Yelick. An Evaluation of Current High-Performance Networks. In *Proceedings IEEE International Parallel & Distributed Processing Symposium (IPDPS '03),* 2003.
9. R. Brightwell, D. Doerfler, K. D. Underwood. A Preliminary Analysis of the InfiniPath and XD1 Interfaces. In *Proceedings IEEE International Parallel & Distributed Processing Symposium (IPDPS '06),* 2006.